**Amendments to the specification:**

**The paragraph at page 1, the line 4 to page 2 line 6:**

Program debugging is done in one of the following two ways. According to a first approach, a debugger is used, a debugger being a tool that enables a partial execution of the program, stopping at predefined points such as lines of code or variable values. A second way is log-based debugging, wherein print statements are added to the program. When a test is executed a log is created. This log is examined off-line by the programmer to look for places where the behavior of the program deviates from the expected behavior. There also exist debugging tools that display traces of executions and show what happens during the execution. Once the location of a problem is found using these tools, other tools such as a debugger are used to debug the program. It is not possible, after identifying the location of the problem, to return to the actual program to investigate the state of the program. Getting to the correct location using a debugger can be a difficult problem, because algorithmic debugging, i.e. locating a bug, is difficult once a fault occurs. Algorithmic debugging is used to support locating a defect once a fault is found, as described in ~~web site http://www.cs.nmsu.edu/~mikau/aadebug.html and~~ "*Generalized algorithmic debugging and testing*" by Peter Fritzson *et al.* appearing in "*ACM Letters on programming languages and testing*", 1:303-322, 1992.

**The paragraph at Page 2, lines 16-27:**

To this end there is provided in accordance with a first aspect of the invention a computer-implemented method for automatically invoking ~~a~~ at least one predetermined debugger command at a desired location of a single thread of a program containing at least one thread, said method comprising:

**(a)** embedding within said single thread at said desired location thereof a utility which reads a trace file in which said at least one predetermined debugger command has been previously embedded; and

A2

**(b)** running the program so that on reaching said desired location, the utility is invoked for reading said trace file and invoking said at least one predetermined debugger command.

**The paragraph at Page 3, lines 17 to 34:**

~~(a)~~ **(i)** an instrumentation scheme, which allows the user to put specialized print statements in the program, both manually and automatically, to create trace files.

~~(b)~~ **(ii)** a modified debugger that can be executed "against" the trace files. This debugger, when encountering an instrumentation statement, will check the trace file. If the trace file contains a debugger command (such as show current program status) it will execute it, else it will continue if appropriate. This is the appearance to the user. Implementation may be done without modifying the debugger.

A3

~~(c)~~ **(iii)** a number of supporting algorithms used to re-execute the program with the same multi-threading interleaving, and, if needed, to create a naming scheme for threads or to match between threads and traces.

**The paragraph at Page 6, lines 11-14:**

When the program is executed on a test, a flag called "**compare**" is set by the **Trace-Print** function according to whether or not the trace file exists on entry to the **Trace-Print** function. If the trace file does not exist, then **compare** is set *false* and the trace file is created otherwise it is set to *true*. ~~added.~~ During subsequent processing, if ~~If~~ **compare**=*false*, then the **Trace-Print** function prints Content into file **Trace-Name** in the form "**Trace: Content**".

A4